Logic-Centred Architecture for Ubiquitous Health Monitoring

Jacek Lewandowski, Hisbel E. Arochena, Raouf N. G. Naguib, Kuo-Ming Chao, Alexeis Garcia-Perez

Abstract:
One of the key points to maintain and boost research and development in the area of Smart Wearable Systems (SWS) is the development of integrated architectures for intelligent services, as well as wearable systems and devices for health and wellness management. This paper presents such a generic architecture for multi-parametric, intelligent and ubiquitous wireless sensing platforms. It is a transparent, smartphone-based sensing framework with customisable wireless interfaces and plug'n'play capability to easily interconnect third party sensor devices. It caters for Wireless Body (BAN), Personal (PAN) and Near-me (NAN) Area Networks. A pivotal part of the platform is the integrated inference engine/runtime environment that allows the mobile device to serve as a user-adaptable personal health assistant. The novelty of this system lays in a rapid visual development and remote deployment model. The complementary visual Inference Engine Editor that comes with the package enables Artificial Intelligence specialists, alongside with medical experts, to build data processing models by assembling different components and instantly deploying them (remotely) on patient mobile devices. In this paper the new logic-centred software architecture for ubiquitous health monitoring applications is described, followed by a discussion as to how it helps to shift focus from software and hardware development, to medical and health process-centred design of new SWS applications.

***Index Terms* — Body sensor networks, Ubiquitous computing, Remote monitoring, Telemedicine, Artificial Intelligence**

I. Introduction

ACCORDING to a recent report published by the World Health Organization [1], well-developed countries are expected to face major challenges in the way current health care services are deployed and delivered. This is due mainly to 1) an aging population, 2) increased life expectancy, and 3) population growth. The United Nations report [2] states that this trend is global and over 60 years old are expected to account for 32% of the population in year 2050. These factors will have a significant impact on the high-rising costs of healthcare liabilities which will eventually outpaced the growth of the overall economy. In response to this we need a more accurate, pre-hospital and prevention-oriented health care system, which will take care of a person's physical health status at its earliest stage, through physical activity management, status monitoring and assessment, as well as early notification in case of an emergency.

One of the possible contributing solutions to these problems could be the implementation of an accurate and ubiquitous health monitoring of individuals by means of smart wearable systems (SWS). SWS are defined as end-to-end, sensor-based integrated systems, capable of sensing, processing, and communicating medical data to interested parties, such as the medical professionals and emergency services, or store it for further reference. It contends to be the next generation e-health systems, delivering patient-oriented services with the vision of empowered health care on the move [3].

In this paper, a new logic-centered software architecture for such ubiquitous health monitoring applications is described. It consists of a transparent, smartphone based sensing framework with customisable wireless interfaces and plug'n'play capability to easily interconnect third party sensor devices in BAN, PAN and NAN Area Networks. A pivotal part of the platform is the integrated inference engine/runtime environment that allows the mobile device to serve as a user-adaptable personal health assistant. The novelty of this system lays on a rapid visual development and remote deployment model. The complementary visual Inference Engine Editor enables machine learning experts along with medical experts to build data processing models by interlacing together different components and controlling the application logic with scripts. The editor allows the instant deployment of such models remotely on patient mobile devices. This approach shifts focus from complex software and hardware development, to simple medical and health process design, what helps to speed up development and deployment of new medical and health applications.

In the following two sections the challenges in SWS adoption as well as state of the art in the field are discussed. Section 4 is dedicated to the formulation of use cases and system requirements, while Section 5 describes the system architecture. In Section 6 the proposed software design is presented. The inference engine, the complementary editor as well as the development and deployment model that they promote are presented in Section 7. Section 8 provides conclusions and future work.

II. Challenges in SWS adoption

Wireless medical telemetry is not a new concept, yet its adoption is minimal in nearly every country. Chan et al. [4] summarised issues preventing the wider acceptance of current Smart Wearable Systems as, amongst others:

- lack of systems' efficiency, reliability, and unobtrusiveness [5-6],
- complexity of system development and validation [7],
- lack of unified multi-platform telemedicine solution for the mobile and desktop operating systems [8],
- not clear requirements from health care professionals and end-users [4],
- cost [9],
- services availability and interoperability issues [10].

As a means to maintain and boost SWS research and development, many researchers [7, 11-12] identified a development of integrated architectures for intelligent home services with wearable systems and devices for home comfort, health and wellness. They concluded that there was currently no smart wearable system on the market integrating several biosensors, intelligent processing and alerts to support medical applications. Such a state of affairs is due to the lack of end-to-end interoperability standards within the sensor networks and also between SWS and disparate healthcare systems. This prevents seamless medical data collection, increases the cost of the systems and their upgrade capabilities, and also limits the shift to systems that are semantically interoperable, process-related, decision-supportive, context-sensitive, user oriented, and trustworthy [13].

III. Related Works

A variety of wireless personal vital signs monitors, both for medical and fitness purposes, are either already on the market, or under development at prototype stage. A full list of wearable systems developed in recent years along with a brief description of their applications can be found in [4]. A vast number of those projects focused at on-body sensing technologies through integration of micro-nano technologies and flexible systems in textile material. They aimed at the implementation of the "e-textile" paradigm, where sensing, actuating, communicating, processing and power sourcing are seamlessly integrated on a textile.

Whereas sensors and actuators are essential to promote SWS adoption amongst the population, they are only means of data collection. The true benefits of health monitoring systems come with data processing and integration. These early systems, were often the side effect of sensor development, generally designed to 'cut the cord' between the patient and the medical professionals, providing mainly only instantaneous single-parameter assessment and transmission. Hence, in order to fully explore the benefits offered by SWS, current research efforts in this area focus on integration and interoperability aspects as well as new classification algorithms [26] which will further boost SWS's adoption and release their commercial value.

In an attempt to design a general-purpose, flexible wireless remote monitoring framework, the noteworthy example of a fully integrated system architecture that took all relevant parties and services on-board, was outlined by Otto et al [14]. Their proposed model spanned a three tier network made of a) tier 1 - Wireless Body Area Network (WBAN), b) tier 2 - individual health monitoring mobile phone system and c) tier 3 - Wide Area Network (WAN) connection to medical servers. The first, fully commercialised product implementing a similar model is MobiHealth [15]. This system provides an integrated mobile remote monitoring and feedback system that integrates with compact third-party sensor systems. Despite having intelligent capabilities to analyse acquired data locally, the main aim of this system was to ensure that patients stayed securely connected to their remote care professional. Moreover, the system introduced for the first time the concept of the M-health service layer, which integrated the intra-BAN and extra-BAN communication, making applications independent from specific characteristics of the underlying communication protocols. This concept of sensor virtualisation and reusable mobile-centric, wireless sensing platform was further developed by the Nokia Remote Sensing (NORS) project [16]. The NORS platform aimed at exploiting the artificial intelligence in several ubiquitous devices that connect locally to sensors and remotely with servers. Depending on the network availability and/or scenario of use, the system allowed users to select where the data processing would take place – locally on the sensor/phone or remotely on the server.

These, as well as other cross platform developments and integration efforts opened new paths in deploying intelligence on distributed devices which informed our model design.

IV. System Requirements

Reviewing previous implementations and their outcomes, several basic functional requirements common to almost every reported SWS system have been identified. These include: a) sensing and filtering, b) data aggregation, c) wireless communication, d) power management, e) data presentation and f) storage. In most cases all these elements are necessary just to get simple sensor readings. To assemble a complete wireless sensors network (WSN) monitoring system traditionally one requires skills in electronics, software engineering, signal processing, control theory, wireless networking and artificial intelligence (AI), to name a few. This, in turn, involves extensive and often platform targeted implementations when, in fact, all what differentiates one application from the other are the sensors used and the data processing algorithms implemented.

This observation suggests that a higher level application developement paradigm could potenatially be applied to SWS systems development which could result in a shift from application development to customisation. This is possible with the use of framework applications, middlewares, runtime environments, scripts and XML. Such platform should offer predefined methods and

paradigms, where design efforts focuses on logic and processes rather than on data aqusition or aggregation. In doing so, it must futher enable:

- Integration of vendor specific sensor nodes under one framework.
- Integration of different wireless communication technologies under one framework.
- Integration of a real-time inference engine, such as artificial neural networks (ANN).
- Customizable context-aware data sampling and efficient data sources utilisation.
- Remote control over sensor nodes through customisable WSN commands.
- Customisable data aggregation from sensor nodes.
- Service oriented design enabling integration with third party web services over WAN connection, where it can be either a service subscriber or a publisher.

Finally, the user should be able to choose whether the system will work as a standalone personal health assistant or as part of a broader telemedicine application.

V. System Architecture

The proposed high-level system architecture (Figure 1) is based on a well established three-tier architecture of the WSN network as proposed in [14], spanning over a network of medical sensors and remote web services. The WSN tier comprises a number of sensor nodes, each capable of sampling, filtering, processing, and communicating physiological signals. The WAN network tier encompasses external web services that can either publish their services or subscribe to the available sensor data sources. The middleware, called personal server, links these tiers together and it is deployed on a smartphone device that interfaces WSN nodes locally and WAN services externally. Moreover it provides integrated sensor nodes management, data aggregation, real-time data processing and transmission, as well as inference capabilities.

The main architectural difference of our proposed model, compared to a typical health monitoring system, lays in the workload distribution, which in terms of data processing, network management, and inference algorithms, is a responsibility of a personal server and sensor nodes, rather than of a remote centralised server. The proposed model allows to eliminate the central medical server from this architecture and instead, dynamic allocate resources and external services [17]. An important advantage of such a two tier model is the improved response time, which is achieved by locating the processing power close to the user, improving therefore user's mobility. Another important advantage achieved is the adaptability aspect, enabling such system to become a user tailored device which can be sensitive to individual's special conditions or behaviours. It also allows to develop algorithms, which will determine the user's state and well-being status ubiquitously, taking into account contextual and patient specific information.

The pivotal element of this system, the AI runtime environment, allows the loading and running of new custom classification and decision algorithms developed in the corresponding Inference Engine Editor. The algorithm can be either downloaded by the user from medical and health process repositories, or can be directly uploaded on the personal server and executed there by health professionals with access rights to it. Such repositories are designed based on the concept of digital distribution platforms for mobile devices, commonly known as application stores.

Reduction of service maintenance costs is one of the most important benefits that come out of ubiquities logic-centered approach to development and more autonomous wellness monitoring systems, such as intelligent Personal Health Assistants (PHA), that require no or only little human intervention. Distributed processing, opposed to only centralised server processing, not only decrease the data transmission cost but can also improve the accuracy of monitoring through patient adaptation, response time and availability of the service to the user. With logic centred development methodology we can "shorten the time-to-market" for new solutions/applications, improve code reusability, reuse existing infrastructure of third party measurement devices but foremost focus on medical and health data processing models what is the future of SWS.

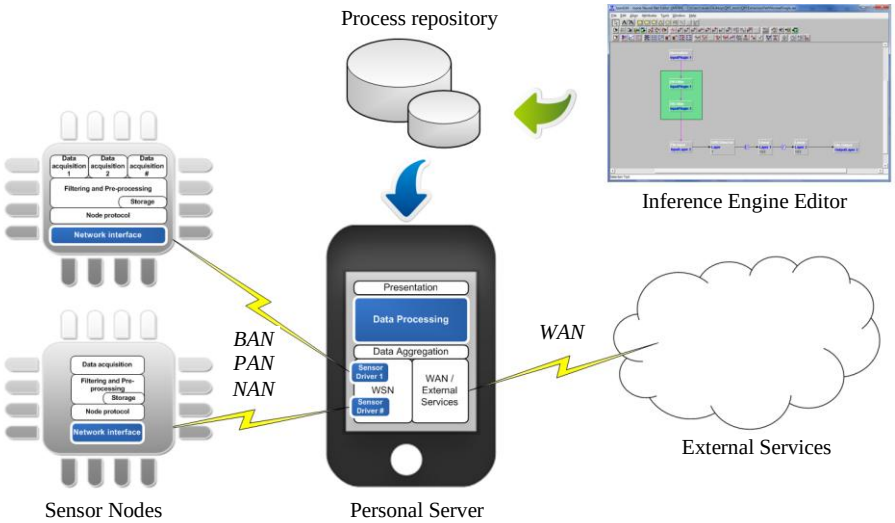


Fig. 1. High level system architecture.

VI. Framework Middlewares

A. Personal server middleware

The personal server middleware consists of a number of specialised software packages which are grouped into two functional layers: data aggregation and data processing accompanied by data presentation layer as illustrated in Figure 2a.

1) Data aggregation layer

The data aggregation layer consist of:

- a node and network interface package,
- the sensor networks abstraction model,
- a WAN/external services package,
- a data acquisition control package

The data aggregation layer is mainly responsible for sensor's data fusion as well as WSN and WAN connections configuration and management. It provides an abstraction and virtualisation of nodes, sensors and connections, through the network manager, sensor manager and WAN services coordinator modules. The abstraction use node specific protocol drivers what makes applications independent from specific characteristics of the underlying communication protocols. It enables customisable wireless interfaces and plug'n'play capability to easily interconnect multiple third party sensor devices and services in BAN, PAN, NAN and WAN networks landscape.

The primary function of this layer is to keep a register of every single data source and manage them accordingly, including discovery, registration, configuration and initialisation. Once the network connections are set up, the data acquisition control package manages the network utilisation, taking care of channel sharing, time synchronisation, data transmission and data encryption. As a result, the data aggregation layer provides the complete collection of real-time, pre-processed and cleaned sensor's data streams ready for processing.

2) Data processing layer

The data processing layer builds upon the data aggregation layer. It focuses on real-time classification and implementation of decision algorithms, applied to the data supplied by the sensor nodes. It consists of:

- data representation package,
- data sources control package,
- data analyses package,
- data storage and distribution packages.

The data processing is based on an inference engine deployed as the data analyses package. It uses the data representation package in order to obtain the higher level semantic data or indexes used for analyses. The inference engine is built based on the Java Object Oriented Neural Engine (JOONE) [18]. This model features a modular architecture made of linkable components that can be used to build not only neural network architectures, but also other types of machine learning algorithms such as Self Organising Maps (SOM), or Support Vector Machine (SVM), amongst others. Each machine learning model is composed of a number of connected components. Depending on how these components are connected, a variety of architectures can be created.

Data processing is capable of taking control over data acquisition through the data sources control package (a decision making tool for data sources management). It introduces the dynamic sensors model which utilises only those sensor channels necessary for accurate system operation. A decision is made based on the decision matrix and decision trees encoding the expert knowledge for outcomes from the data analyses module. Decisions take the form of actions such as: a) to add/remove a new sensor channel for more accurate monitoring, b) to use external data services, or c) to reconfigure the current data sources.

Other predefined blocks include the data distribution mechanism, which posts alarms and notification remotely to third parties, and a file system to storage monitoring logs.

B. Sensor Node Middleware

The Sensor node middleware consists of components that sample, filter and process physiological signals. Such data is then stored locally or transmitted to the personal server middleware for integration, analysis and decision making.

The prototype software runs on the TinyOS platform. The applications are implemented as a set of component modules written in nesC. A prototype sensor node middleware has been developed paying special attention to the reusability, flexibility and customization of its components (Figure 2b) following design patterns presented in [19]. With this in mind, our proposed application architecture consists of the following components:

- Sensor component (SensorC and interface Sense), responsible for data sampling on analog-to-digital converter's inputs, implemented using the Facade pattern which defines a coherent abstraction boundary by exporting the interfaces of several sub components.
- Filter component (Filter#C and Filter interface), responsible for signal filtering, implemented using Service Instance pattern which provide multiple instances of a particular service sharing the same code;
- Processor component (Processor#C and Data interface), responsible for data pre-processing, implemented using the Decorator pattern enhances the capabilities and functionality of the SensorsC and Filter#C components without modifying their implementation.
- Storage component (StorageC and interface Store), responsible for local data storage on flash memory, implemented using the Facade pattern which allows for a single configuration that simplifies dependency resolution.
- Node Protocol component (NodeProtocolC that provides Protocol interface as well as its subsequent Operation#C components and interface Operation), responsible for performing an externally customizable set of operations in response to the input from the network interface or call from App module, implemented using the Dispatcher pattern.
- Communication Stack component (BTCommStackC and interface CommStack), which implements the network interface responsible for data transmission, in this case using Bluetooth radio. This functionality is

a) b)

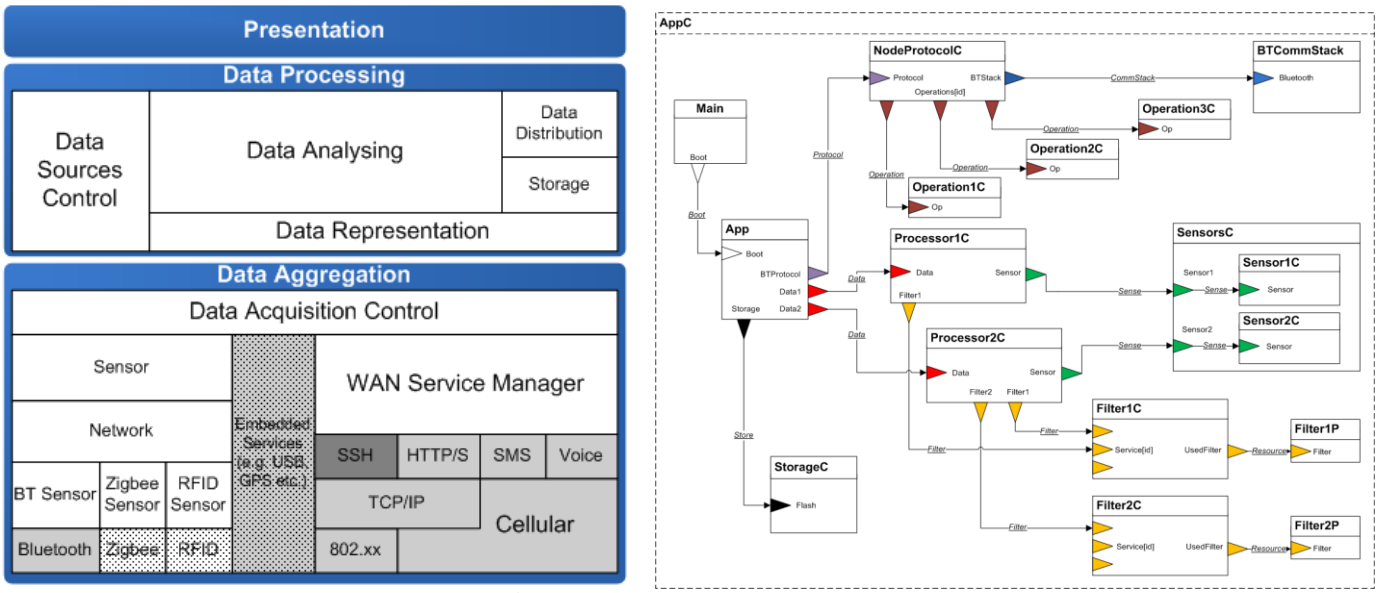


Fig. 2. a) Block diagram of personal server middleware, b) TinyOS sensor node middleware.

encapsulated in the Adapter pattern which converts the protocol specific interface into a single interface type CommStack, what simplifies access to the network resources.

VII. AI Runtime and Inference Engine Editor

The biggest advantage of the proposed logic centered architecture lays on the embedded inference engine, which serves as an artificial runtime environment for the algorithms and constitutes the central component of the system. It offers capabilities to allow the development of new machine learning algorithms, and instantly deploys them remotely on the user mobile device without needing to modify or re-implement the whole application again.

A complementary visual Inference Engine Editor, offered with the package, enables AI experts and health professionals to build new inference models for their applications in a very short time. This is done by linking together components and, in cases where out-of-the–box implementations are needed, the application logic can be controlled with scripts. Depending on the nature of the problem any type of algorithm(s) can be used to solve it. Each model can be composed of one or many components of different types. All components are pluggable, reusable, parameterisable and serialisable code modules. These features guarantee the scalability, reliability and expansibility of the model, enabling the feature selection, training, validation and verification, data dimensionality reduction and testing of complex hybrid models in one editor. The resulting algorithm converts to a self contained object by serialising the network as a byte stream to the file system or database. Such file can then be sent to the remote mobile device using a WAN connection and resurrected there. Deployment of the algorithm on mobile devices can be accomplished by embedding the algorithm into a custom framework mobile application.

A framework application, such as our personal server, has to then supply data to the algorithm inputs, interrogate these, and read the results on its output. There are two ways to accomplish this, depending on how the embedding application interrogates the algorithm it can choose between synchronous or asynchronous mode. In synchronous mode, the algorithm is interrogated by the application by setting its input patterns and calling a run method. After processing the results are stored in memory and it waits to be collected by the application for further processing or presentation. In asynchronous mode, the algorithm runs as a background process, and an external asynchronous source of data interrogates the algorithm with an input pattern as it arrives. Such source of data can be a sensor device or an external data acquisition service. The example of such development and deployment process for the simple neural network algorithm illustrates Figure 3.

VIII. Evaluation

Based on the system architecture presented above, we have rapidly developed a prototype vital signs monitoring system to continuously monitor and analyse five vital signs and their trends using two sensor nodes. The first sensor node is a chest strap, capable to measure ECG, temperature, and respiratory rate, which use off-the-shelf SHIMMER wireless sensor platform [20] with number of sensors such as SHIMMER ECG daughter card, NTC type thermistor and piezoelectric sensor. It has been programmed using our TinyOS sensor node middleware. The second sensor node used in our case study is a commercially available wireless pulse oximeter, which was rapidly integrated with the aid of the personal server framework. The prototype of the personal server has been implemented for CLDC 1.1 and MIDP 2.0 profiles in Java programming language. The prototype system was tested on phoneME Java Virtual Machine [40] that can run on Windows, IOS and Android.

Our objective was to assess qualitatively the effectiveness of the proposed architecture in enabling a flexible design of algorithms and clean application implementation. This was measured in terms of code

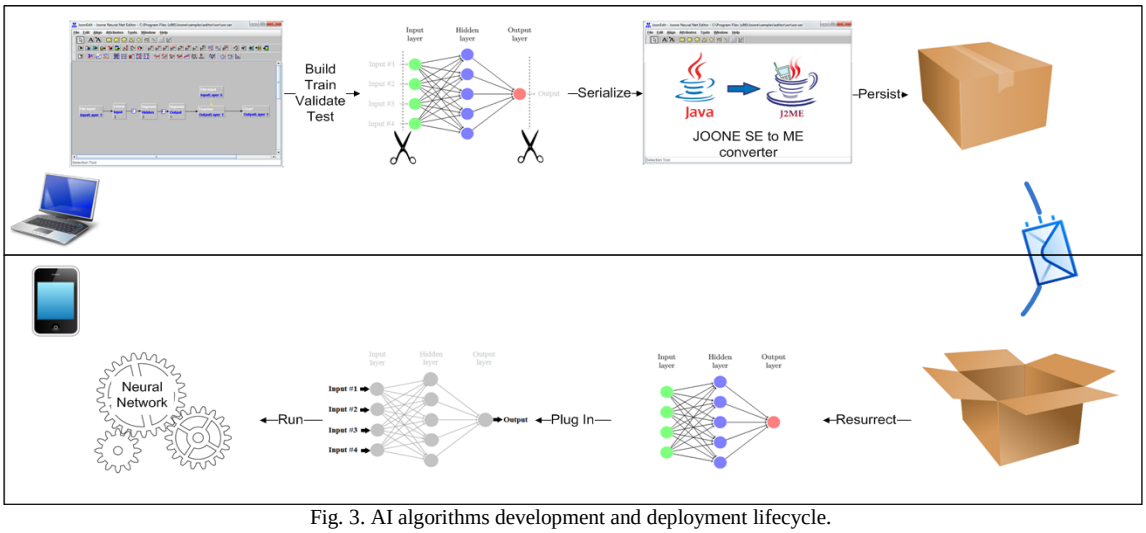


Fig. 3. AI algorithms development and deployment lifecycle.

complexity, scalability, heterogeneity and code reuse, while preserving such performance measures as memory requirements, power saving and network throughput.

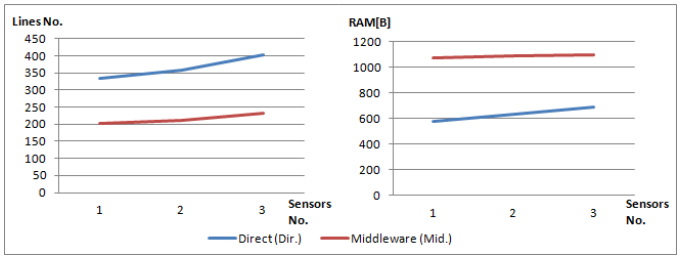
The design process, as well as performance of applications using Sensor Node middleware, were compared with their direct implementation on top of TinyOS. For such purposes (1) AccelECG application, available from TinyOS contributed code repository and its two extensions (2) AccelECG_TEMP and (3) AccelECG _TEMP_RESP, were re-implemented using the proposed sensor node middleware. Main qualitative issues that manifest the proposed sensor node middleware to be superior over the direct implementation are:

- Direct TinyOS implementation is much more complex where developers must manually wire all subsequent components, implement their interfaces, control message transmission, parsing, buffering data etc. while using the middleware the focus is on the application specific processing related to the actual data of interest with all configuration being done a prior by the model.
- The middleware enables the network heterogeneity of the application in terms of communication protocol and its sensitivity to network conditions what in case of direct implementation is equivalent to app redesign.
- The middleware improves the scalability of the application in case when new sensors must be deployed on the node, enabling to reuse existing code by adding new Sensor component with its specific processing.
- Data flow in the TinyOS application is not evident due to split phase operation, what greatly complicate the maintenance and debugging. The proposed middleware alleviates this problem with the Processor component.

Quantitative measures used to compare the complexity of those applications are presented in Figure 4 below. The comparison includes the number of lines of code, which for this examples decreased by almost 50% with use of the middleware. Also using the middleware the number of explicit application events that the developer must control to simply get sensor readings decreased to 4 events which are independent from the number of sensors. These events include: connection made, command received, sensing done, connection closed. Such simplification comes at the price of a slight increase in the size of the binary code deployed on the mode (ROM) as well as heap size (RAM). This, however, remains well within the limits of program flash memory size of the commercially available sensor platforms (e.g. 48KB ROM + 10KB RAM for SHIMMER).

Qualitative evaluation of the Personal Server and Inference Engine conducted during the application design process reveled model inherent properties such as:

- Heterogeneity of the middleware that manifest in its ability to easily integrate multiple third party sensor devices and communication protocols was obtained through the sensor abstraction model, which simplifies data acquisition and network coordination by hiding the whole network specific functionality, what enables to perceived sensors as a data publishing service within any type of network.
- Scalability achieved with modified Flooding Time Synchronization Protocol (FTSP) with dynamic slot allocation for efficient use of the available bandwidth what increased the node sleep time and helped to reduce the transmission and energy cost.
- Usability, enabled by the high-level abstraction of application design process using embedded inference engine and its rapid deployment model facilitated by the visual data flow composition using visual editor.



App.	Lines of code		Explicit events		ROM [KB]		RAM [B]	
	Dir.	*Mid.*	*Dir.*	*Mid.*	*Dir.*	*Mid.*	*Dir.*	*Mid.*
(1)	334	201	8	4	8.96	10.87	576	1074
(2)	357	212	10	4	9.26	10.91	634	1086
(3)	402	231	12	4	9.87	10.96	690	1094

Fig. 4. Comparison of middleware against direct TinyOS implementation.

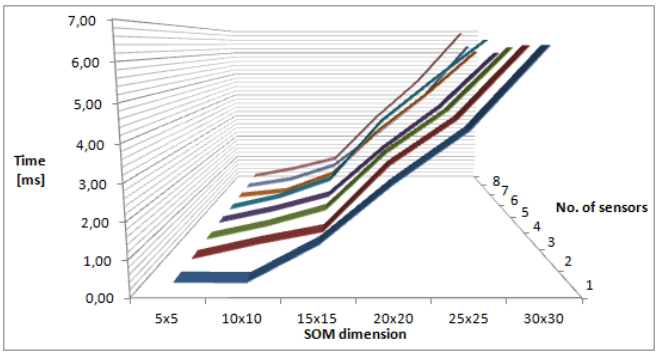


Fig. 5. Impact of number sensors and complexity of inference algorithms on sample execution time

With a corresponding visual editor and training environment, a subject specific user such as medical professional is able to build, train, test and validate the algorithm, and subsequently, in a very short period of time, conduct initial field tests.

To verify that the advantages we identified above do not negatively affect the system performance, we extended our evaluation beyond the development model to look at how the network overheads and inference engine model affect the sample execution time. As expected, Figure 5 shows significant correlation between execution time and the complexity of the inference algorithm (measured by the Self Organizing Map dimension). However, it also shows that the network overheads associated with the increase of number of sensor data streams does not affect the execution time what proves the quality of the presented model.

IX. Conclusions and Future Work

The main innovations presented in this paper are: 1) the newly proposed concept of logic-centred design methodology for ubiquitous health monitoring enabled by the presented integrative SWS architecture; and 2) the embedded inference engine model that can be hosted locally on the personal server. Both contributes to shift focus from software and hardware development to medical and health process-centered design when developing new smart wearable systems. Thanks to presented capabilities, a normally extensive implementation project can be achieved in a limited time frame and reduced to components assembly and configuration.

Future work will focus on porting and extending further the platform to support any native smartphone operating system, what will translate on an improvement on the adoption rate of systems build using this framework. It is envisaged that this can be achieved with a central meta code repository, which at the time of deployment, will port the application to the right platform following methods presented by Miroslav et al. [8] Other areas of future work are development of further test beds with applications offering solutions to various health monitoring problems. By enabling the acquisition of different body parameters from commercially available third party devices as well as integration with wider telemedicine systems and external services, we aim to boost the adoption and integration of SWS in everyday life.

References

[1] World Health Organisation, "The world health report 2010: Health systems financing: the path to universal coverage," World Health Organisation, Geneva2010.

[2] United Nations, "Population Ageing and Development 2012 (Wall Chart)," Population Division, Department of Economic and Social Affairs, New York, NY, ISBN 978-92-1-151494-0, 2012.

[3] R. S. H. Istepanian, S. Laxminarayan, and C. S. Pattichis, *M-Health: Emerging Mobile Health Systems*: Springer US, 2006.

[4] M. Chan, D. Estève, J.-Y. Fourniols, C. Escriba, and E. Campo, "Smart wearable systems: Current status and future challenges," *Artificial Intelligence in Medicine*, vol. 56, pp. 137-156, 2012.

[5] B. K. Hensel, G. Demiris, and K. L. Courtney, "Defining Obtrusiveness in Home Telehealth Technologies: A Conceptual Framework," *Journal of the American Medical Informatics Association*, vol. 13, pp. 428-431, July 1, 2006 2006.

[6] J. Ko, C. Lu, M. B. Srivastava, J. A. Stankovic, A. Terzis, and M. Welsh, "Wireless Sensor Networks for Healthcare," *Proceedings of the IEEE*, vol. 98, pp. 1947-1960, 2010.

[7] A. Lymberis and A. Dittmar, "Advanced Wearable Health Systems and Applications - Research and Development Efforts in the European Union," *Engineering in Medicine and Biology Magazine, IEEE*, vol. 26, pp. 29-33, 2007.

[8] K. Miroslav, L. Fedor, and V. Gabriel, "Multi-platform telemedicine system for patient health monitoring," in *Biomedical and Health Informatics (BHI), 2012 IEEE-EMBS International Conference on*, 2012, pp. 127-130.

[9] T. S. Bergmo, "Economic evaluation in telemedicine – still room for improvement," *Journal of Telemedicine and Telecare*, vol. 16, pp. 229-231, July 1, 2010 2010.

[10] R. S. H. Istepanian, E. Jovanov, and Y. T. Zhang, "Guest Editorial Introduction to the Special Section on M-Health: Beyond Seamless Mobility and Global Wireless Health-Care Connectivity," *Information Technology in Biomedicine, IEEE Transactions on*, vol. 8, pp. 405-414, 2004.

[11] I. Korhonen, J. Parkka, and M. Van Gils, "Health monitoring in the home of the future," *Engineering in Medicine and Biology Magazine, IEEE*, vol. 22, pp. 66-73, 2003.

[12] P. Świątek, P. Stelmach, A. Prusiewicz, and K. Juszczyszyn, "Service Composition in Knowledge-based SOA Systems," *New Generation Computing*, vol. 30, pp. 165-188, 2012/06/01 2012.

[13] B. G. Blobel, "Educational challenge of health information systems' interoperability," *Methods of Information in Medicine*, vol. 46, pp. 52-56, 2007.

[14] C. Otto, A. Milenkovic, C. Sanders, and J. E., "System architecture of a wireless body area sensor network for ubiquitous health monitoring," *Mobile Multimedia Journal*, vol. 1, pp. 307-326, 2006.

[15] A. van Halteren, R. Bults, K. Wac, N. Dokovsky, G. Koprinkov, I. Widya, *et al.*, "Wireless body area networks for healthcare: the MobiHealth project," *Stud Health Technol Inform*, vol. 108, pp. 181-93, 2004.

[16] D. Trossen and D. Pavel, "Building a ubiquitous platform for remote sensing using smartphones," in *Mobile and Ubiquitous Systems: Networking and Services, 2005. MobiQuitous 2005. The Second Annual International Conference on*, 2005, pp. 485-489.

[17] A. Grzech, P. Świątek, and P. Rygielski, "Dynamic Resources Allocation for Delivery of Personalized Services," in *Software Services for e-World*. vol. 341, W. Cellary and E. Estevez, Eds., ed: Springer Berlin Heidelberg, 2010, pp. 17-28.

[18] P. Marrone. (2007). *JOONE: The Complete Guide*. Available: http://www.joone.org

[19] D. Gay, P. Levis, and D. Culler, "Software design patterns for TinyOS," *ACM Trans. Embed. Comput. Syst.*, vol. 6, p. 22, 2007.

[20] M. J. McGrath and T. J. Dishongh, "A Common Personal Health Research Platform—SHIMMER and BioMOBIUS," *Intel Technology Journal*, vol. 13, pp. 122-147, 2009.